PATENT ABSTRACTS OF JAPAN

(11)Publication number : 05-197547
(43)Date of publication of application : 06.08.1993

(51)Int.Cl. G06F 9/38
G06F 9/38

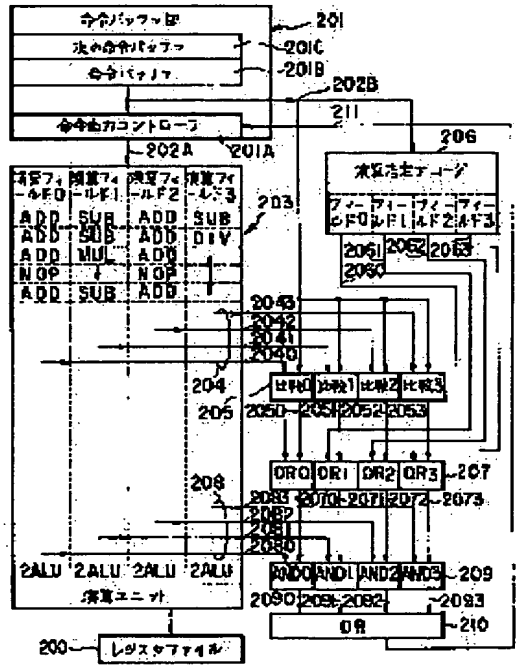
(21)Application number : 04-155190
(22)Date of filing : 15.06.1992

(71)Applicant : TOSHIBA CORP
(72)Inventor : ISHIKAWA TEI

(30)Priority
Priority number : 03286100 Priority date : 31.10.1991 Priority country : JP
03286224 31.10.1991 JP

(54) VLIW TYPE ARITHMETIC PROCESSOR

(57)Abstract:
PURPOSE: To speed up arithmetic processing by executing following instruction words in parallel without waiting the completion of the arithmetic of precedent instruction words unless a register number is referred to for the arithmetic of the following instruction words.
CONSTITUTION: A comparator 205 compares the number of a register expected to store the execution result of an instruction, which begins to be executed earlier than an instruction word to currently be executed, with the number of a register to be read out by the instruction to currently be executed. When there is a matching register number, the instruction to currently be executed is held in a waiting state by an instruction controller 201A once it is confirmed that the contents of the register having the register number can not be used yet. In other cases, the following instructions which do not refer to the execution result of the instruction being executed are executed in parallel even while the instruction which requires plural cycles for its processing is being executed, thus speed up the processing of plural instruction words.



LEGAL STATUS

[Date of request for examination] 09.03.1998

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]	2928684
[Date of registration]	14.05.1999
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平5－197547

(43)公開日　平成５年(1993)８月６日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　9/38	３１０　Ｊ	9290－5B		
	３７０　Ｘ	9290－5B		

審査請求　未請求　請求項の数１（全 15 頁）

(21)出願番号　特願平4－155190

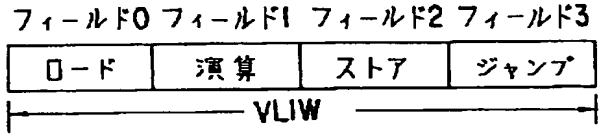
(22)出願日　平成４年(1992)６月15日

(31)優先権主張番号　特願平3－286100
(32)優先日　平３(1991)10月31日
(33)優先権主張国　日本（ＪＰ）
(31)優先権主張番号　特願平3－286224
(32)優先日　平３(1991)10月31日
(33)優先権主張国　日本（ＪＰ）

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地
(72)発明者　石川　禎
東京都府中市東芝町１番地　株式会社東芝府中工場内
(74)代理人　弁理士　鈴江　武彦

(54)【発明の名称】　ＶＬＩＷ型演算処理装置

(57)【要約】
【目的】ＶＬＩＷ型演算処理装置における複数命令並列処理の速度改善。
【構成】先行する演算が実行中のため内容が確定していないレジスタ（オペランド）を後続の命令語が参照しようとする場合、あるいは後続命令語がまだ演算実行中の演算器を使用しようとする場合には、演算が完了して演算結果が確定するまで後続命令語の実行を待たせ、これらの場合以外では先行する命令語の演算が実行中であっても後続命令語を実行する制御を導入して、並列処理の効率を改善する。



【特許請求の範囲】
【請求項１】　複数の命令語を含むＶＬＩＷを格納する格納手段と；前記ＶＬＩＷに含まれる命令語数に対応した数の複数フィールドからなるパイプラインを有し、これらのパイプラインで前記ＶＬＩＷに含まれる命令を独立に実行する実行手段と；前記実行手段により実行される各命令語のオペランドにより参照されるハードウエアリソースと；前記実行手段のあるフィールドで現在実行中の命令の実行結果を格納するための第１オペランドと、前記実行手段の各フィールドで現在実行しようとする命令により参照される第２オペランドとを比較し、前記第２オペランドが前記第１オペランドと一致するものを含む場合に、一致信号を提供する提供手段と；前記実行手段の各フィールドで命令実行が終了していない場合に、フィールド毎に未了信号を発生する発生手段と；前記提供手段からの前記一致信号と前記発生手段からの前記未了信号との論理積を前記実行手段の各フィールド毎にとり、何れかのフィールドが前記論理積に対して真をもたらす場合に命令取り込み禁止信号を出力し、全てのフィールドが前記論理積に対して疑をもたらす場合に命令取り込み信号を出力する出力手段と；前記出力手段が前記命令取り込み信号を出力するときは前記格納手段から前記実行手段の前記パイプラインへ前記ＶＬＩＷに含まれる１以上の命令語を同時に転送し、前記出力手段が前記命令取り込み禁止信号を出力するときは前記格納手段から前記実行手段への前記ＶＬＩＷの命令語転送を停止する手段とを具備したことを特徴とするＶＬＩＷ型演算処理装置。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】この発明は、ストアドプログラム型デジタル計算機におけるＶＬＩＷ（Ｖｅｒｙ　Ｌｏｎｇ　Ｉｎｓｔｒｕｃｔｉｏｎ　Ｗｏｒｄ）型演算処理装置に関する。
【０００２】
【従来の技術】マイクロプロセサの処理性能は、その動作周波数、１命令当たりの所要サイクル数ＣＰＩ（Ｃｙｃｌｅｓ　ｐｅｒ　Ｉｎｓｔｒｕｃｔｉｏｎ）、およびプログラム実行に必要な命令数の３つの要因で決まる。すなわち、処理性能を上げたければ、動作周波数を上げ、ＣＰＩを下げ、プログラム実行命令数を減らすことが必要となる。
【０００３】上記３要素のうち、半導体製造技術の進歩によりマイクロプロセサチップの動作周波数は年々上がっているが、その時々で入手可能なチップの動作周波数には上限がある。
【０００４】また、マイクロプロセサにはＣＩＳＣ（Ｃｏｍｐｌｅｘｅｄ　Ｉｎｓｔｒｕｃｔｉｏｎ　Ｓｅｔ　Ｃｏｍｐｕｔｅｒ）型とＲＩＳＣ（Ｒｅｄｕｃｅｄ　Ｉｎｓｔｒｕｃｔｉｏｎ　Ｓｅｔ　Ｃｏｍｐｕｔｅｒ）型（およびこれらの折衷型）があり、ＲＩＳＣチップならＣＩＳＣチップよりもＣＰＩを小さくできるが、それでもＣＰＩは１までにしか下がらない。プログラム実行命令数はプログラマの能力（あるいはコンパイラの性能）に依存するが、これを減らすにも限界がある。
【０００５】このような状況を背景として、ＶＬＩＷアーキテクチャとスーパースカラアーキテクチャが生まれてきた。これらのアーキテクチャを採用したプロセサでは、プログラム中から並列に実行できる命令を見つけて、複数の処理を並列に実行する。すなわち、複数の命令を同時にデコードしてそれらを実行ユニットに送り、同時に処理を行なう。このような並列処理により、ＣＰＩを１以下に下げることが可能となる。
【０００６】ＶＬＩＷ型プロセサおよびスーパースカラ型プロセサは複数のパイプラインを内臓しており、このパイプラインにより複数の処理を同時に実行できる点では同じであるが、両者は以下の点で相違する。
（１）命令ストリームのスケジューリング方法の相違
【０００７】スーパースカラでは、プロセサの命令デコーダが並列に実行できる命令を見つけ、命令実行段階で動的にスケジューリングを行なう。このため、並列性の検出はフェッチした命令からしか行なえない。
【０００８】これに対し、ＶＬＩＷでは、コンパイル時（あるいはアセンブラコーディング時）にプログラムをスケジューリングする。そのため、スーパースカラに比べコンパイラ（あるいはプログラマ）の負担は大きくなるが、ＶＬＩＷではハードウエア（命令デコーダ）が簡単になる。
（２）命令フォーマットの相違
【０００９】一般に、スーパースカラは従来のＲＩＳＣプロセサと同じ３２ビット長の命令フォーマットを持っている。そのデコーダは、フェッチした命令から並列に実行できる命令を選び、パイプラインに投入する。３２ビット長の命令フォーマットをサポートするスーパースカラプロセサでは従来のＲＩＳＣプロセサの命令フォーマットをそのまま利用することができ、オブジェクトコードレベルで、既存のＲＩＳＣとソフトウエアの互換性を維持できる。
【００１０】これに対し、ＶＬＩＷは、一般に３２ビットよりも長い命令フォーマットを持っており、１命令で複数の処理を指定することができる。そのため、既存のＲＩＳＣとソフトウエア互換性を維持することはできないが、スーパースカラよりも高い性能を出すことができる。
【００１１】ところで、パイプライン構造を持つプロセサの場合、１つの命令は複数のステージに分けられて実行される。１つのステージは原則として１クロックで実行されるため、パイプラインが順調に動作すれば全ての命令は見かけ上１クロックで実行されることになる。しかし、実際には様々な原因でパイプラインは乱れプロセ

サの実質的な処理速度は低下する。これらの原因はハザードと呼ばれる。このハザードをハードウエアで検出して自動的にパイプラインを遅らせる機構をインターロックという。

【００１２】この発明は、スーパースカラではなくＶＬＩＷアーキテクチャに基づく並列処理システムに関するものである。とくに、上記インターロックの機能を改良し、パイプラインの遅れを最小限に押さえ処理の高速化を図ったＶＬＩＷ型演算処理装置に関する。

【００１３】一般に、ＶＬＩＷ型演算処理装置では複数の演算を指定できる固定長の命令語を実行して行くが、１つの命令語中に指定された複数の演算は、少なくともソフトウエア（プログラム）からは同時に実行されているように見えることが必要である。

【００１４】ＶＬＩＷ型演算処理装置では、その処理能力の向上のために浮動小数点演算などもサポートされている。この浮動小数点演算は、種類によっては（例えば除算では）その処理に数サイクルを要する。

【００１５】ところが、従来のＶＬＩＷ型演算処理装置では、同時に実行されるべく指定された１命令中の複数演算は、ソフトウエアに対しては、あくまで同時に実行されるように見せなければならない。そのために、同時に指定された複数演算中に処理時間が長くかかるものがあった場合、それ以外の演算が先に終了していたとしても、この長時間演算処理が完了するまでは、演算が終了した演算器に次の演算処理をさせず、遊ばせていた。

【００１６】

【発明が解決しようとする課題】上述のように、同時に実行されるべく指定された１命令語中の複数演算はソフトウエア上はあくまで同時に実行されていることになっていなければならないために、従来のＶＬＩＷ型演算処理装置では、先行する命令語に含まれる演算処理が全て終了するまでは次の命令語を実行しないようにしている。このため、同時に指定された複数演算中に処理時間の長くかかるものがあった場合、それが完了するまでは、他の演算が既に終了していたとしても、演算の終了している演算器に対して次の命令を実行させずに待たせたまま遊ばせることになる。すると並列演算時間が長くかかり、ＶＬＩＷアーキテクチャの並列演算処理能力を十分に活用できない。

【００１７】この発明は、上記事情に鑑みなされたもので、ソフトウエアに対しては演算の実行順序を保証しつつ、先行して起動された時間のかかる演算が実行中であっても、できるかぎり後続の命令語を並列に実行できるようにし、もって並列処理用演算器の稼働率を向上させ、演算処理装置の全体的な処理能力を改善したＶＬＩＷ型演算処理装置を提供することを目的とする。

【００１８】

【課題を解決するための手段】この発明のＶＬＩＷ型演算処理装置は、複数の命令語を含むＶＬＩＷを格納する命令バッファ（２０１）と；

【００１９】前記ＶＬＩＷに含まれる命令語数に対応した数の複数フィールドからなるパイプラインを有し、これらのパイプラインで前記ＶＬＩＷに含まれる命令を独立に実行する演算ユニット（２０３）と；前記演算ユニットにより実行される各命令語のオペランド（Ｒｊ）により参照されるハードウエアリソース（２００）と；

【００２０】前記演算ユニットのあるフィールド（ｉ＝１）で現在実行中の命令（ＭＵＬ）の実行結果を格納するための第１オペランド（Ｒ４）と、前記演算ユニットの各フィールド（ｉ＝０～３）で現在実行しようとする命令（Ａｄｄ；ｉ＝０，２）により参照されるの第２オペランド（Ｒ１～Ｒ４）とを比較し、前記第２オペランド（Ｒ１～Ｒ４）が前記第１オペランド（Ｒ４）と一致するもの（Ｒ４）を含む場合に、一致信号を提供する比較器（２０５）と；前記演算ユニットの各フィールド（ｉ）で命令実行が終了していない場合に、フィールド毎に未了信号を発生する手段（２０８）と；

【００２１】前記比較器からの前記一致信号と前記発生手段からの前記未了信号との論理積を前記演算ユニットの各フィールド（ｉ）毎にとり、何れかのフィールド（ｉ）が前記論理積に対して真をもたらす場合に命令取り込み禁止信号を出力し、全てのフィールド（ｉ）が前記論理積に対して疑をもたらす場合に命令取り込み信号を出力する論理回路（２０９，２１０）と；

【００２２】前記論理回路が前記命令取り込み信号を出力するときは前記命令バッファから前記演算ユニットの前記パイプラインへ前記ＶＬＩＷに含まれる１以上の命令語を同時に転送し、前記論理回路が前記命令取り込み禁止信号を出力するときは前記命令バッファから前記演算ユニットへの前記ＶＬＩＷの命令語転送を停止する命令出力コントローラ（２０１Ａ）とを具備している。

【００２３】

【作用】この発明のＶＬＩＷ型演算処理装置では、現在実行しようとしている命令語より先に実行が開始された命令が実行中であり、その命令の実行結果を格納する予定のレジスタ番号と現在実行しようとしている命令によって読出されるレジスタ番号とを比較する。そして一致するレジスタ番号がある場合に当該レジスタ番号のレジスタの内容がまだ使用不可能であることが認識されると、現在実行しようとしている命令を待たせる。それ以外の場合では、処理に複数サイクルかかる命令が実行中であっても、その命令の実行結果を参照しない後続の命令は並列実行させて、複数の命令語の処理の高速化を図る。

【００２４】

【実施例】図１は、フィールド０～３それぞれに４つの３２ビット長命令語（ロード、演算、ストア、ジャンプ）を配した１２８ビットのＶＬＩＷの一例を示す。この演算命令には、加算（ＡＤＤ）、減算（ＳＵＢ）、乗

算（ＭＵＬ）、除算（ＤＩＶ）、無処理（ＮＯＰ）等が含まれる。

【００２５】図２は、この発明が適用されない場合のＶＬＩＷ並列演算処理の一例を示す。この例では、フィールド３で除算（Ｒ３／Ｒ４）を４クロックかけて実行中にフィールド０～２の実行ユニットが３クロック分遊んでしまっており、フィールド１で乗算（Ｒ１＊Ｒ２）を２クロックかけて実行中にフィールド０，２，３の実行ユニットが１クロック分遊んでしまっている。この遊びがあるために、図２で例示する演算処理の完了には、都合８クロックかかっている。

【００２６】図３は、この発明が適用された場合のＶＬＩＷ並列演算処理の一例を示す。この例では、フィールド３で４クロック除算（Ｒ３／Ｒ４）を実行中に、フィールド１で２クロック乗算（Ｒ１＊Ｒ２）と２度の１クロック減算（Ｒ１－Ｒ２）を実行し、フィールド０とフィールド２で２度の１クロック加算（Ｒ１＋Ｒ２；Ｒ３＋Ｒ４）とタイミング合わせのための１クロックＮＯＰを１度実行している。図３に例示する演算処理では、除算・乗算の処理中にパイプライン中の実行ユニットが遊ぶ（ＮＯＰ実行）割合が少ないので、都合５クロックで図２と同等の演算処理を終了している。図６は、図３で例示したような並列演算処理がハードウエア中でどのように進行するかを説明するフローチャートである。

【００２７】まず、例えば図３の命令語１がハードウエア（並列演算実行ユニット）にフェッチされたとする（ＳＴ１００）。このフェッチにより、フィールド０～３の４本のパイプラインには命令語１のＡＤＤ，ＳＵＢ，ＡＤＤ，ＳＵＢが一括して（同期して）投入される。

【００２８】各フィールド０～３では、投入された命令（ＡＤＤ，ＳＵＢ，ＡＤＤ，ＳＵＢ）のオペランド（命令語が参照するアドレスまたはデータのこと。ここでは各パイプラインで参照されるレジスタファイルのレジスタ番号Ｒｊ；ｊは例えば１～８）が使用可能かどうかチェックされる（ＳＴ１１０～ＳＴ１１３）。今は演算を開始し始めたばかりであり、投入された命令（ＡＤＤ，ＳＵＢ，ＡＤＤ，ＳＵＢ）が要求する各フィールド０～３のオペランド（Ｒ１～Ｒ４）は全て使用可能である（ＳＴ１１４～ＳＴ１１７，イエス）。

【００２９】フィールド０～３のオペランドが全て使用可能となると（ＳＴ１２０，イエス）、各フィールド０～３の演算に必要なハードウエアリソース（レジスタファイル中で現在使用可能なレジスタ）があるかどうかチェックされる（ＳＴ１３０～ＳＴ１３３）。今は演算を開始し始めたばかりであり、ハードウエアのリソースは全て使用可能である（ＳＴ１３４～ＳＴ１３７，イエス）。

【００３０】フィールド０～３のハードウエアリソースが全て使用可能となれば（ＳＴ１４０，イエス）、各フィールド０～３で、投入された命令（ＡＤＤ，ＳＵＢ，ＡＤＤ，ＳＵＢ）の処理が同時に開始する（ＳＴ１５０～ＳＴ１５３）。以上の処理により、図３の命令語１に対する処理が１クロックで完了する。

【００３１】次に、図３の命令語２がハードウエアにフェッチされる（ＳＴ１００）。このフェッチにより、フィールド０～３の４本のパイプラインには命令語２のＡＤＤ，ＳＵＢ，ＡＤＤ，ＤＩＶが投入される。

【００３２】各フィールド０～３では、投入された命令（ＡＤＤ，ＳＵＢ，ＡＤＤ，ＤＩＶ）のオペランドが使用可能かどうかチェックされる（ＳＴ１１０～ＳＴ１１３）。命令語１は１クロックで同時処理完了しており、かつフィールド３の命令ＤＩＶが参照するオペランド（レジスタＲ８）はまだ未使用なので、投入された命令（ＡＤＤ，ＳＵＢ，ＡＤＤ，ＤＩＶ）が要求する各フィールド０～３のオペランド（Ｒ１～Ｒ４，Ｒ８）は全て使用可能である（ＳＴ１１４～ＳＴ１１７，イエス）。

【００３３】フィールド０～３のオペランドが全て使用可能となると（ＳＴ１２０，イエス）、各フィールド０～３の演算に必要なハードウエアリソースがあるかどうかチェックされる（ＳＴ１３０～ＳＴ１３３）。投入された命令（ＡＤＤ，ＳＵＢ，ＡＤＤ，ＤＩＶ）が要求するハードウエアのリソースはいずれも他の命令により使用中でないので、全て使用可能である（ＳＴ１３４～ＳＴ１３７，イエス）。

【００３４】フィールド０～３のハードウエアリソースが全て使用可能となれば（ＳＴ１４０，イエス）、各フィールド０～３で、投入された命令（ＡＤＤ，ＳＵＢ，ＡＤＤ，ＤＩＶ）の処理が同時に開始する（ＳＴ１５０～ＳＴ１５３）。以上のようにして、図３の命令語２が処理される。

【００３５】次に、図３の命令語３がハードウエアにフェッチされる（ＳＴ１００）。このフェッチにより、フィールド０～２の３本のパイプラインには命令語３のＡＤＤ，ＭＵＬ，ＡＤＤが投入される。（図３の処理プログラムを生成したコンパイラは、命令語２でフィールド３に投入した命令ＤＩＶが時間のかかる処理であることを知っているため、この時点ではフィールド３に新たな命令は投入されない。）

【００３６】各フィールド０～２では、投入された命令（ＡＤＤ，ＭＵＬ，ＡＤＤ）のオペランドが使用可能かどうかチェックされる（ＳＴ１１０～ＳＴ１１３）。ここで、フィールド３はまだ命令語２のＤＩＶを処理中であり、この命令ＤＩＶの処理はフィールド３の２つのＡＬＵ内で行なわれている。このため、フィールド３のオペランド（Ｒ３，Ｒ４，Ｒ８）はまだ塞がっておらず、使用可能である（ＳＴ１１７，イエス）。また、命令語２のフィールド０～２は１クロックで同時処理完了しているので、投入された命令（ＡＤＤ，ＭＵＬ，ＡＤＤ）が要求する各フィールド０～２のオペランド（Ｒ１～Ｒ

４）も全て使用可能である（ＳＴ１１４～ＳＴ１１６，イエス）。

【００３７】フィールド０～３のオペランドが全て使用可能となると（ＳＴ１２０，イエス）、各フィールド０～３の演算に必要なハードウエアリソースがあるかどうかチェックされる（ＳＴ１３０～ＳＴ１３３）。

【００３８】この場合、フィールド１で処理中の命令ＭＵＬはリソース（レジスタＲ４）を使っており、このリソース（レジスタＲ４）をフィールド２の命令ＡＤＤが参照しているため、フィールド２の演算に必要なハードウエアリソースはない（ＳＴ１３６，ノー）。そこで、このリソース（レジスタＲ４）が空くまで（つまりフィールド１の命令ＭＵＬの実行が終わるまで）、フィールド２は何もしない命令ＮＯＰを挟んで待つ（ＳＴ１３２，ＳＴ１３６のループ）。

【００３９】このとき、フィールド２の命令ＡＤＤはフィールド０の命令ＡＤＤが使うリソース（レジスタＲ３）も参照している。このため、フィールド１の命令ＭＵＬの実行が終わるまでフィールド２が命令ＮＯＰ（１クロック）を実行して待っている間、フィールド０も命令ＮＯＰ（１クロック）を実行して待つ（ＳＴ１３０，ＳＴ１３４のループ）。

【００４０】フィールド０、２に命令ＮＯＰを挟むことによりフィールド０～３のハードウエアリソースが全て使用可能となり（ＳＴ１４０，イエス）、各フィールド０～３で、投入された命令（ＡＤＤ，ＭＵＬ，ＡＤＤ）および実行中の命令（ＤＩＶ）の並列処理が、２クロックかけて同時進行する（ＳＴ１５０～ＳＴ１５３）。以上のようにして、図３の命令語３が処理される。

【００４１】次に、図３の命令語４がハードウエアにフェッチされる（ＳＴ１００）。このフェッチにより、フィールド０～２の３本のパイプラインには命令語４のＡＤＤ，ＳＵＢ，ＡＤＤが投入される。（図３の処理プログラムを生成したコンパイラは、命令語２でフィールド３に投入した命令ＤＩＶが時間の係る処理であることを知っているため、ここでもフィールド３に新たな命令は投入されない。）

【００４２】各フィールド０～２では、投入された命令（ＡＤＤ，ＳＵＢ，ＡＤＤ）のオペランドが使用可能かどうかチェックされる（ＳＴ１１０～ＳＴ１１３）。命令語３のフィールド０～２は２クロックで同時処理完了しているので、投入された命令（ＡＤＤ，ＳＵＢ，ＡＤＤ）が要求する各フィールド０～２のオペランド（Ｒ１～Ｒ４）は全て使用可能である（ＳＴ１１４～ＳＴ１１６，イエス）。また、フィールド３はまだ命令語２のＤＩＶをフィールド３のＡＬＵで処理中であり、この命令ＤＩＶに対するオペランド（Ｒ３，Ｒ４，Ｒ８）は使用可能である（ＳＴ１１７，イエス）。

【００４３】フィールド０～３のオペランドが全て使用可能となると（ＳＴ１２０，イエス）、各フィールド０～３の演算に必要なハードウエアリソースがあるかどうかチェックされる（ＳＴ１３０～ＳＴ１３３）。投入された命令（ＡＤＤ，ＳＵＢ，ＡＤＤ）および実行中の命令（ＤＩＶ）が要求するハードウエアのリソースは塞がっていないので、全て使用可能である（ＳＴ１３４～ＳＴ１３７，イエス）。

【００４４】フィールド０～３のハードウエアリソースが全て使用可能となれば（ＳＴ１４０，イエス）、各フィールド０～３で、投入された命令（ＡＤＤ，ＳＵＢ，ＡＤＤ）および実行中の命令（ＤＩＶ）の並列処理が同時進行する（ＳＴ１５０～ＳＴ１５３）。以上のようにして、図３の命令語４が処理される。

【００４５】図７は、この発明の一実施例に係るＶＬＩＷ並列演算処理装置の構成を説明するブロック図である。この実施例装置は、レジスタファイル２００と、命令バッファ部２０１と、演算ユニット２０３と、レジスタ比較回路２０５と、演算指定デコーダ２０６と、オア回路２０７と、アンド回路２０９と、オア回路２１０とを備えている。

【００４６】命令バッファ部２０１は、図１に示すような複数演算を包含する命令語群からなるＶＬＩＷを格納する。命令バッファ部２０１は、後述する信号２１１によって命令バッファ２０１Ｂ内のＶＬＩＷを出力する命令出力コントローラ２０１Ａと、命令バッファ２０１Ｂ内の命令の次に実行される命令を格納するバッファ２０１Ｃを含んでいる。

【００４７】命令バッファ部２０１に格納されたＶＬＩＷは、信号線２０２Ａを介して演算ユニット２０３に転送される。演算ユニット２０３は、転送されたＶＬＩＷに含まれる複数演算を、レジスタファイル２００を用いて、４本のパイプライン（フィールド０～３）により並列処理する。この処理のために、これらのパイプラインはそれぞれ独自の演算器（ＡＬＵ）を複数台（例えば２台）備えている。

【００４８】演算ユニット２０３で現在実行中の演算の結果が格納されるところのレジスタファイル２００のレジスタ番号Ｒｊ（ｊは例えば１～８）は、信号線２０４ｉ（ｉ＝０，１，２，３）を介して、レジスタ比較回路２０５に与えられる。このレジスタ比較回路２０５には、信号線２０２Ｂを介して、次に実行しようとするＶＬＩＷによって参照されるレジスタ番号も与えられる。レジスタ比較回路２０５は、前者のレジスタ番号と後者のレジスタ番号をと比較し、両者が一致するかどうかを調べる。

【００４９】すなわち、レジスタ比較回路２０５は４つの比較ブロック０～３により構成され、これらのブロック０～３は演算ユニット２０３のフィールド０～３にそれぞれ対応している。各ブロックｉ（ｉ＝０～３）では、そのフィールドｉで現在実行中の演算の結果が格納されるレジスタ番号Ｒｊ（ｊ＝１～８）と、次の命令語

によって参照される全てのレジスタ番号との一致が調べられる。

【００５０】演算指定デコーダ２０６には、命令バッファ２０２Ｂから、信号線２０２Ｂを介して、次に実行しようとしている命令語が与えられる。デコーダ２０６は与えられた命令語から、ＮＯＰ以外の演算が指定されているフィールドｉを検出する。そのようなフィールドｉが検出されたなら、デコーダ２０６はこのフィールドｉの演算器が空くまで次の命令語の実行を待たせる信号２０６ｉを出力する。

【００５１】オア回路２０７は、レジスタ比較回路２０５の各ブロックｉごとの出力２０５ｉと演算指定デコーダ２０６の各フィールドｉごとの出力２０６ｉとの論理和を取る。またアンド回路２０９は、オア回路２０７の各ＯＲゲートｉごとの出力２０７ｉと信号２０８ｉとの論理和を取る。信号２０８ｉは、演算ユニット２０３の各フィールドｉから取り出されるもので、各フィールドｉごとに現在実行中の演算がまだ完了していないことを示す。

【００５２】アンド回路２０９の各ＡＮＤゲートｉごとの出力２０９ｉはオア回路２１０に入力される。オア回路２１０の出力は、信号線２１１を介して、命令出力コントローラ２０１Ａに送られる。

【００５３】次に、図８および図９を参照して、上記構成のＶＬＩＷ型演算処理装置の動作を説明する。（この動作はマイクロプログラムではなくハードウエアロジックにより実行される。）

【００５４】この実施例のＶＬＩＷ型演算処理装置は、次のような特徴を持つ。すなわち、原則として複数命令語の並列処理を可とするが、現在実行中の命令語の演算結果が格納されるレジスタファイル２００のレジスタ番号（オペランド）と、次に実行しようとしている命令語によって参照されるレジスタ番号（オペランド）とが一致し、かつ一致したレジスタ番号のレジスタへ結果を格納する演算がまだ完了していない場合に、命令バッファ部２０１から演算ユニット２０３への命令フェッチを待たせることに特徴を持つ。

【００５５】そこで、命令バッファ部２０１から信号線２０２Ａを介して１つのＶＬＩＷが演算ユニット２０３にフェッチされ（ＳＴ１０）演算処理がなされると、演算ユニット２０３のフィールドｉで現在実行中の演算結果が格納されるレジスタ番号（Ｒｊ）が信号線２０４ｉから取り出され、次に実行しようとしているＶＬＩＷに含まれる全ての命令語によって参照されるレジスタ番号が命令バッファ２０１Ｂの信号線２０２Ｂから取り出される。信号線２０４ｉおよび信号線２０２Ｂから取り出されたレジスタ番号は、比較回路２０５において、一致するかどうかチェックされる。

【００５６】比較回路２０５の４ブロック０～３各々は演算ユニット２０３のフィールド０～３に対応しており、１つのブロックｉではそのフィールドｉで現在実行中の演算結果が格納されるレジスタ番号と次のＶＬＩＷ内の命令語によって参照される全てのレジスタ番号との一致が調べられる。一致があれば（ＳＴ１４，イエス）比較回路２０５の出力２０５ｉは〝１〟となり（ＳＴ１６）、不一致ならば（ＳＴ１４，ノー）出力２０５ｉは〝０〟となる（ＳＴ１８）。

【００５７】比較回路２０５からの一致出力２０５ｉはオア回路２０７の各ＯＲゲート０～３の一方に入力される。各ＯＲゲート０～３の他方には、演算指定デコーダ２０６からの出力２０６ｉが入力される。

【００５８】演算指定デコーダ２０６は、次に実行しようとしている命令語中で、ＮＯＰおよびこの演算が指定されているフィールドｉを検出する（ＳＴ２０）。フィールドｉでＮＯＰが検出されると（ＳＴ２２，イエス）、そのフィールドｉに対応するＯＲゲートにロジック〝０〟の出力２０６ｉが与えられる（ＳＴ２４）。ＮＯＰが検出されなければ（ＳＴ２２，ノー）、出力２０６ｉは〝１〟となる（ＳＴ２６）。

【００５９】出力２０５ｉおよび出力２０６ｉの何れかが〝１〟であれば（ＳＴ２８，イエス）、オア回路２０７の各ＯＲゲート０～３の出力２０７ｉは〝１〟となる（ＳＴ３０）。出力２０７ｉが〝１〟というのは、「次に実行しようとする命令語が、フィールドｉ用のハードウエアリソースを使用する」か、あるいは「次に実行しようとする命令語が、フィールドｉで現在実行中の演算結果を参照する」ということを意味する。この出力２０７ｉはアンド回路２０９の対応するＡＮＤゲートの一方入力に与えられる。

【００６０】一方、フィールドｉで現在実行中の演算がその処理サイクル中でまだ終了していないなら（ＳＴ３２，イエス）、未終了のフィールドｉの出力２０８ｉが〝１〟となる（ＳＴ３４）。この出力２０８ｉはアンド回路２０９の対応するＡＮＤゲートの他方入力に与えられる。

【００６１】フィールドｉについての出力２０７ｉおよび出力２０８ｉがともに〝１〟であれば（ＳＴ３６，イエス）、このフィールドｉに対応するアンド回路２０９のＡＮＤゲートの出力２０９ｉは〝１〟となる（ＳＴ３８）。この出力２０９ｉが〝１〟というのは、「フィールドｉで現在実行中の演算が終了するのを待て」ということを意味する。

【００６２】アンド回路２０９の何れかのＡＮＤゲート０～３の出力２０９ｉが真（ロジック〝１〟）であれば（ＳＴ４０，イエス）、オア回路２１０の出力は真（ロジック〝１〟）となる（ＳＴ４２）。すると、信号線２１１を介して命令バッファ部２０１の出力コントローラ２０１Ａに待ち信号（ロジック〝１〟）が伝達され、命令バッファ部２０１は演算ユニット２０３へのＶＬＩＷの一括転送を停止する。

【００６３】フィールド０～３全ての演算が完了すると出力２０８ｉは全て″０″になる（ＳＴ４４，イエス）。するとオア回路２０７の出力２０７ｉに関係なくアンド回路２０９の出力２０９ｉは全て″０″になるから、信号線２１１の出力も″０″になる（ＳＴ４６）。すると命令出力コントローラ２０１Ａでの命令転送禁止が解除され、バッファ２０１Ｂ内の次のＶＬＩＷが演算ユニット２０３にフェッチされる（ＳＴ４８）。

【００６４】このようにして、図２に示すような命令語１～４が命令バッファ部２０１に格納されていてこれらの命令語を順次実行しようとする場合を想定すると、上記実施例のＶＬＩＷ型演算処理装置では次の処理が実現される。

【００６５】すなわち、命令語２と命令語３と命令語４における演算器の参照関係についてコンパイラレベルで考察すれば、命令語２では２つの加算と１つの減算と１つの除算との４演算を行ない、その演算結果をレジスタＲ１，Ｒ３，Ｒ４，Ｒ８に格納するようにしている。このうち、Ｒ１，Ｒ３，Ｒ４は命令語３で読出されるが、Ｒ８は命令語３、命令語４とも参照しない。このＲ８の中味は除算結果なので、処理に４クロックかかるのではあるが、命令語３と命令語４は必ずしもこの除算の完了を待ち合わせる必要はない。

【００６６】命令語３は、２つの加算と１つの乗算の計３つの演算を実行して、Ｒ１，Ｒ３，Ｒ４に結果を格納する。これらのＲ１，Ｒ３，Ｒ４は命令語４で読出されるのであるが、とくに乗算の結果が格納されるＲ４は命令語４で参照される。このため、命令語４は、命令語３の直後の命令であるが、命令語３の乗算の完了を待ち合わせる必要がある。

【００６７】したがって、図２に示す命令語１～４を順次処理する場合には、図３に示すように、命令語２の除算を実行している最中に命令語３と命令語４を実行することができる。ただし、命令語４は命令語３の乗算結果を参照しているので、命令語３の乗算結果が出るのを待ってから命令語４を実行することになる。この場合、命令語３と命令語４の実行時間は合わせて３クロック分となるが、これは命令語２の除算と並行して実行されるので、命令語２の完了と命令語４の完了は同時になる。

【００６８】こうして、図２に示すような命令語１～４を実行する場合、図３に示すような処理内容と時間経過となる。すなわち、図３では、各命令語１～４の処理時間の総和（所要クロック数）は、図２の場合の８クロックと比べて３クロック少ない５クロックとなる。

【００６９】また、図４に示す命令語１～４を順次処理する場合には、図５に示すような処理内容と時間経過となる。図２の場合では命令語４のフィールド３はＮＯＰであるのに対して、図４に示す命令語４のフィールド３は加算を指定している。その他については、図２の命令語の内容と図４の命令語の内容とは同じである。上記命令語３のフィールド３における内容の違いが処理にどう影響するかについて、以下説明する。

【００７０】演算ユニット２０３では、各フィールド０～３において、加算・減算・乗算・除算・ＮＯＰの何れかの演算を行なうことができるが、同時には１種類の演算しか処理できない。そうであれば、図４の命令語４は、命令語２のフィールド３の除算が完了するまでは実行できないことになる。このため、先行する演算を実行中の演算器を後続する命令が使用しようとする場合には、先行演算が完了するまで後続の演算を待たせる制御が必要がある。すなわち、図４の命令語４の実行は命令語２の除算が完了した後に開始されるべきである。このような制御を基に図４に示す命令語１～４が順次処理されると、図５に示すようになり、命令語１から命令語４までの全実行時間は６クロックとなる。

【００７１】なお、演算ユニット２０３の各フィールド０～３において同時に異なった種類の演算が実行できるような構成であれば、図４の命令語４は命令語２の除算と並行して処理できるので、図４の命令列の全実行時間は５クロックで済むことになる。

【００７２】以上のように、この発明の実施例では、先行する演算が実行中のために内容未確定のレジスタを後続命令語が参照しようとする場合と、まだ演算実行中の演算器を使用しようとする場合には、演算が完了して結果が確定するまで後続命令語の実行を待たせる制御を導入しているが、これらの場合以外では、先行する命令語の演算が実行中であっても、原則としてその演算の完了を待たずに後続命令語が実行できるようにして、並列処理能力を向上させている。

【００７３】図１０は、この発明の他実施例に係るＶＬＩＷ並列演算処理装置を示す。図１０の実施例は、図７の実施例から、次に実行されるＶＬＩＷ中からＮＯＰ命令を検出する構成（演算指定デコーダ２０６）を省略したものとなっている。

【００７４】図１０の構成では、ＶＬＩＷによるオペランドの参照関係をコンパイラレベルで考慮して、以下の制御を行なうことで、フィールド０～３における処理の並列性を確保している。

【００７５】すなわち、ある命令語により参照されるオペランド（レジスタ番号）に演算結果を格納しようとするところの、先行起動された演算（例えば乗算）がまだ完了していない場合は、その完了を待ち、そうでない場合は、まだ処理途中の先行起動演算（例えば除算）があっても、この処理途中の演算と並行して後続命令を実行できるような制御を行なう。

【００７６】これにより、例えば図３において、フィールド３で除算処理途中にその他の演算処理をフィールド０～２で並列に実行できる。換言すると、ＶＬＩＷのオペランドフェッチステージでは全てのフィールドに対して命令語を同期して一括投入するが、それらの命令語の

実行ステージではフィールド毎に独立した並列処理を行なうことにより、各フィールドの演算手段（ＡＬＵ等）の遊びを極力減らす制御が行なわれる。

【００７７】

【発明の効果】以上述べたように、この発明では、先行する命令語の演算結果を格納しようとするレジスタ番号が後続の命令語の演算で参照されない場合（または先行命令語演算のために使用中の演算器が後続命令語演算によって使用されない場合）、先行する命令語の演算完了を待たずに後続命令語の並列実行を行なうよう構成したので、並列処理効率が高く、全体として演算処理の高速化が図れる。

【００７８】さらに、上記構成において、処理に時間のかかる演算の結果を参照する他の演算の命令語が、前者の演算の命令語から極力離れるようなコードを生成するコンパイラを用意すれば、この発明の装置の処理能力をさらに改善することができる。

【図面の簡単な説明】

【図１】図１は、ＶＬＩＷ命令の一例を示す図。

【図２】図２は、この発明が適用されない場合のＶＬＩＷ並列演算処理の一例を説明する図。

【図３】図３は、この発明が適用された場合のＶＬＩＷ並列演算処理の一例を説明する図。

【図４】図４は、この発明が適用されない場合のＶＬＩＷ並列演算処理の他例を説明する図。

【図５】図５は、この発明が適用された場合のＶＬＩＷ並列演算処理の他例を説明する図。

【図６】図６は、この発明のＶＬＩＷ並列演算の処理の流れを例示するフローチャート。

【図７】図７は、この発明の一実施例に係るＶＬＩＷ並列演算処理装置の構成を説明するブロック図。

【図８】図８は、図７の装置によるＶＬＩＷ並列演算処理の動作を説明するフローチャートの一部。

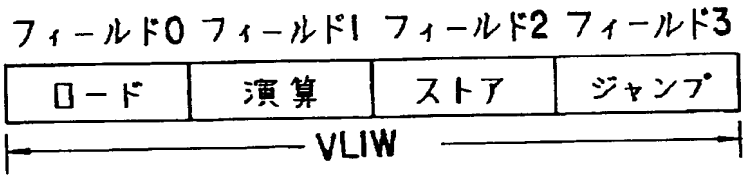
【図９】図９は、図７の装置によるＶＬＩＷ並列演算処理の動作を説明するフローチャートの他部。

【図１０】図７は、この発明の他実施例に係るＶＬＩＷ並列演算処理装置の構成を示すブロック図。

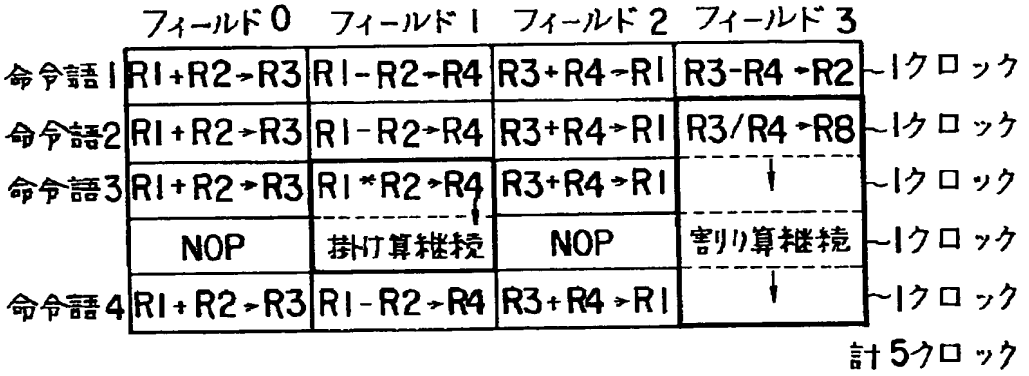
【符号の説明】

２００…レジスタファイル（ハードウエアリソース）、２０１…命令バッファ部（格納手段）、２０１Ａ…命令出力コントローラ（停止手段）、２０１Ｂ，２０１Ｃ…命令バッファ、２０３…演算ユニット（実行手段）、２０５…レジスタ比較回路（提供手段）、２０６…演算指定デコーダ（実行手段）、２０７…ＯＲ回路、２０８…信号線（発生手段）、２０９…ＡＮＤ回路（出力手段）、２１０…ＯＲ回路（出力手段）、２０５ｉ…一致信号、２０８ｉ…未了信号、２１１…命令取り込み信号（″０″）／命令取り込み禁止信号（″１″）。

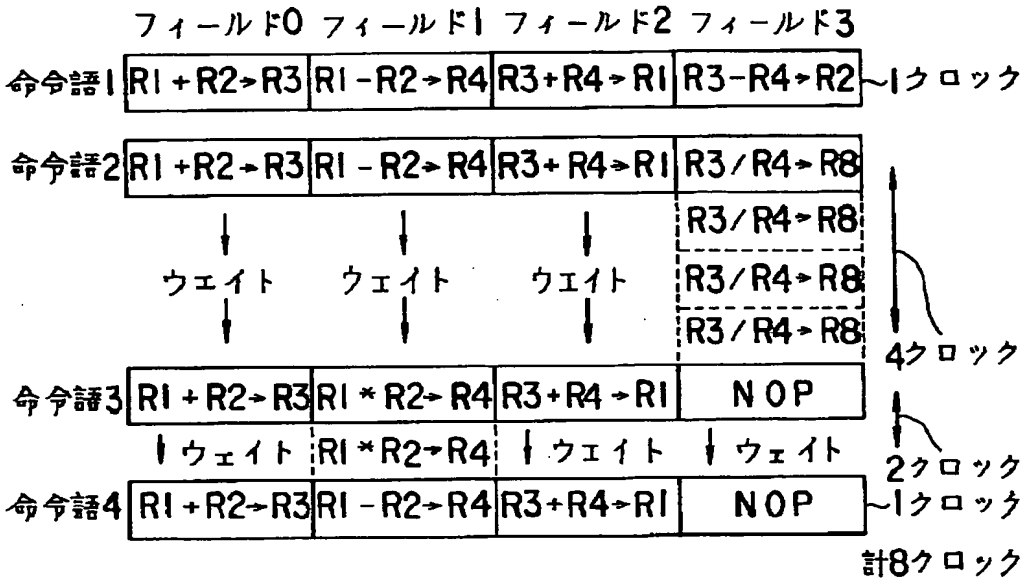
【図１】



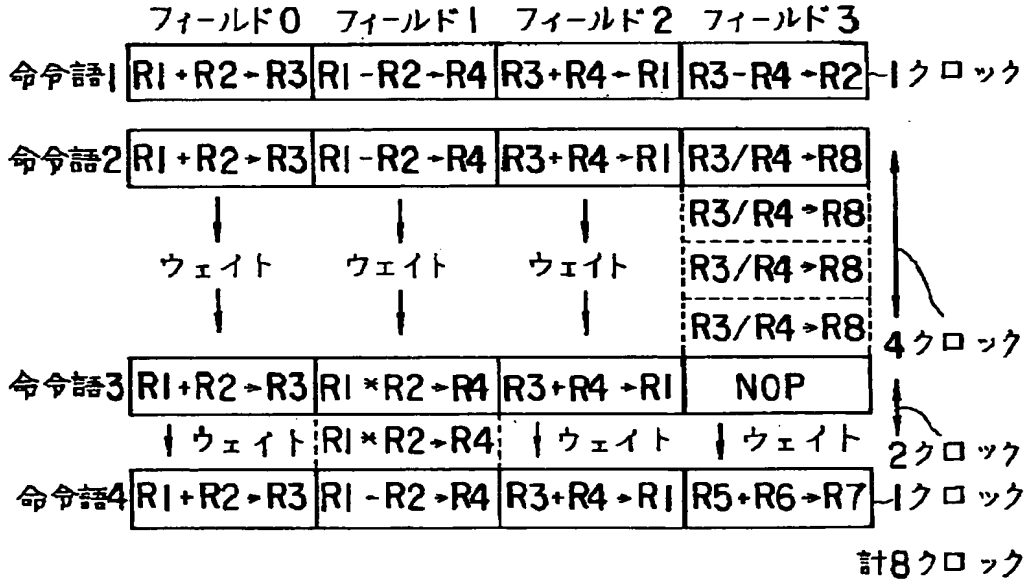
【図３】



【図２】


フィールド0 フィールド1 フィールド2 フィールド3
命令語1
R1+R2→R3
R1-R2→R4
R3+R4→R1
R3-R4→R2
1クロック
命令語2
R1+R2→R3
R1-R2→R4
R3+R4→R1
R3/R4→R8
R3/R4→R8
R3/R4→R8
R3/R4→R8
ウェイト
ウェイト
ウェイト
4クロック
命令語3
R1+R2→R3
R1*R2→R4
R3+R4→R1
NOP
ウェイト
R1*R2→R4
ウェイト
ウェイト
2クロック
命令語4
R1+R2→R3
R1-R2→R4
R3+R4→R1
NOP
1クロック
計8クロック

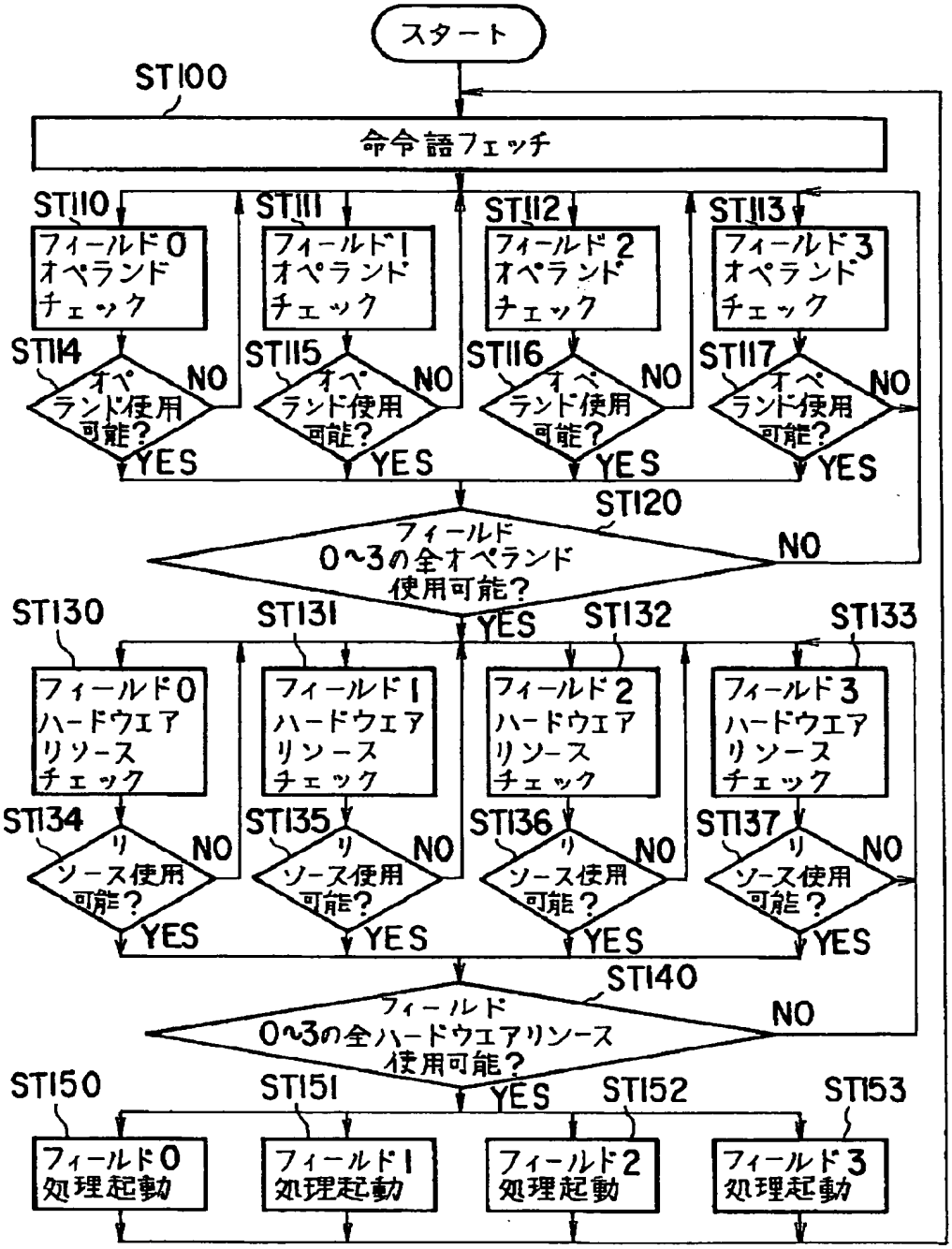
【図４】


フィールド0 フィールド1 フィールド2 フィールド3
命令語1
R1+R2→R3
R1-R2→R4
R3+R4→R1
R3-R4→R2
1クロック
命令語2
R1+R2→R3
R1-R2→R4
R3+R4→R1
R3/R4→R8
R3/R4→R8
R3/R4→R8
R3/R4→R8
ウェイト
ウェイト
ウェイト
4クロック
命令語3
R1+R2→R3
R1*R2→R4
R3+R4→R1
NOP
ウェイト
R1*R2→R4
ウェイト
ウェイト
2クロック
命令語4
R1+R2→R3
R1-R2→R4
R3+R4→R1
R5+R6→R7
1クロック
計8クロック

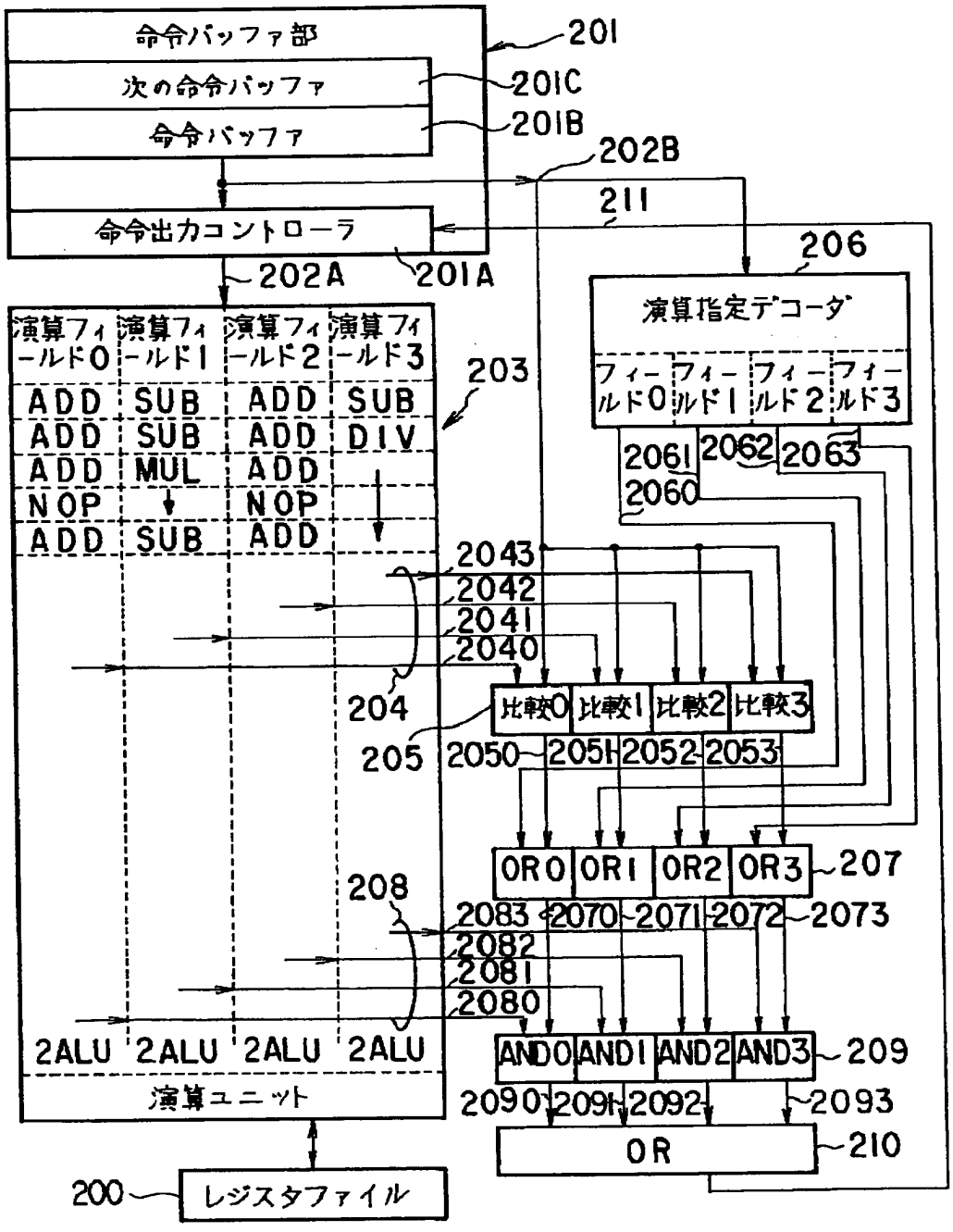
【図５】

	フィールド0	フィールド1	フィールド2	フィールド3	
命令語1	R1＋R2→R3	R1－R2→R4	R3＋R4→R1	R3－R4→R2	～1クロック
命令語2	R1＋R2→R3	R1－R2→R4	R3＋R4→R1	R3／R4→R8	～1クロック
命令語3	R1＋R2→R3	R1＊R2→R4	P3＋R4→R1	↓	～1クロック
	NOP	掛け算継続	NOP	割り算継続	～1クロック
	NOP	NOP	NOP	↓	～1クロック
命令語4	R1＋R2→R3	R1－R2→R4	R3＋R4→R1	R5＋R6→R7	～1クロック
					計6クロック

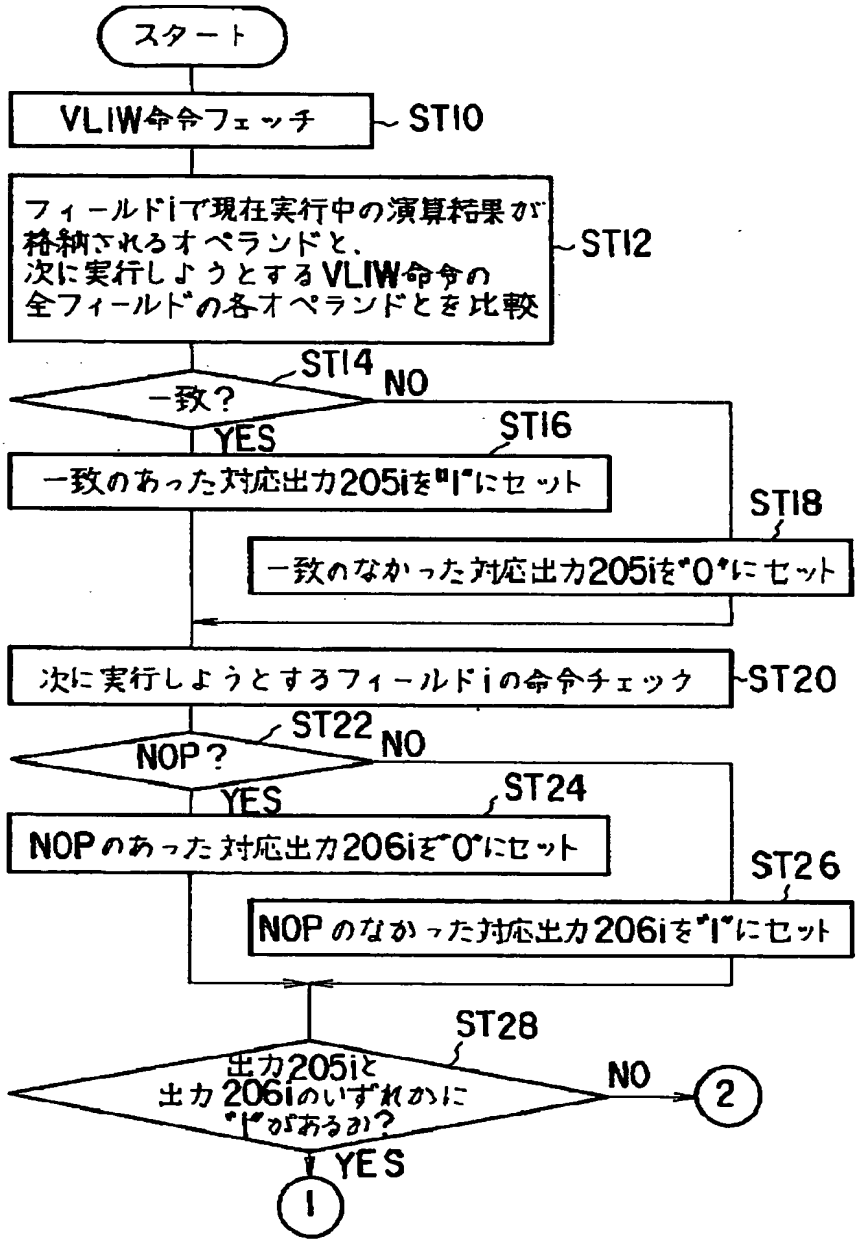
【図６】


スタート
ST100
命令語フェッチ
ST110
フィールド0 オペランド チェック
ST111
フィールド1 オペランド チェック
ST112
フィールド2 オペランド チェック
ST113
フィールド3 オペランド チェック
ST114
オペランド使用可能?
NO
YES
ST115
オペランド使用可能?
NO
YES
ST116
オペランド使用可能?
NO
YES
ST117
オペランド使用可能?
NO
YES
ST120
フィールド 0~3の全オペランド 使用可能?
NO
YES
ST130
フィールド0 ハードウエア リソース チェック
ST131
フィールド1 ハードウエア リソース チェック
ST132
フィールド2 ハードウエア リソース チェック
ST133
フィールド3 ハードウエア リソース チェック
ST134
リソース使用可能?
NO
YES
ST135
リソース使用可能?
NO
YES
ST136
リソース使用可能?
NO
YES
ST137
リソース使用可能?
NO
YES
ST140
フィールド 0~3の全ハードウエアリソース 使用可能?
NO
YES
ST150
フィールド0 処理起動
ST151
フィールド1 処理起動
ST152
フィールド2 処理起動
ST153
フィールド3 処理起動

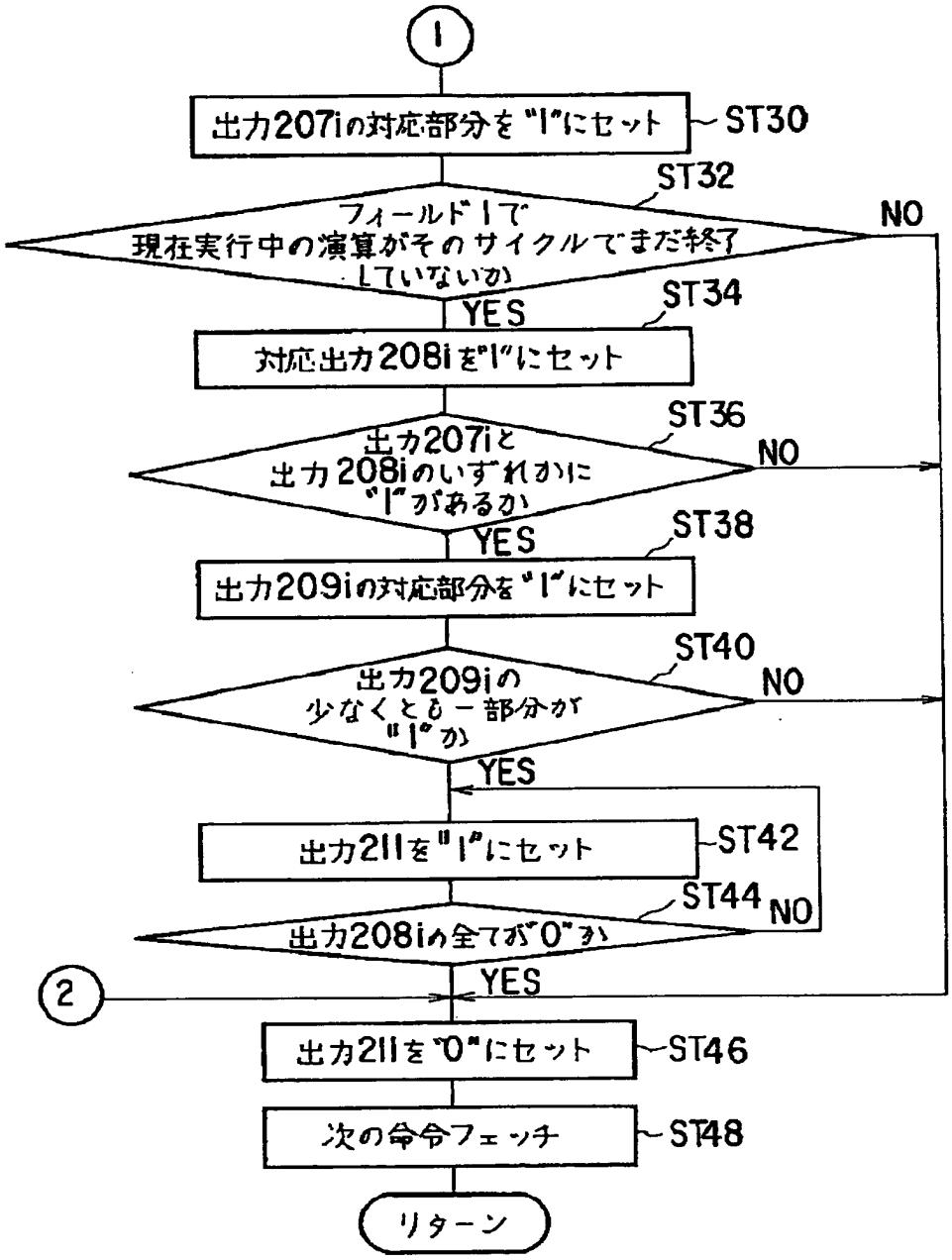
【図7】


命令バッファ部
201
次の命令バッファ
201C
命令バッファ
201B
202B
211
命令出力コントローラ
202A
201A
206
演算指定デコーダ
フィールド0
フィールド1
フィールド2
フィールド3
2061
2062
2063
2060
演算フィールド0
演算フィールド1
演算フィールド2
演算フィールド3
203
ADD SUB ADD SUB
ADD SUB ADD DIV
ADD MUL ADD
NOP NOP
ADD SUB ADD
2043
2042
2041
2040
204
比較0 比較1 比較2 比較3
205
2050 2051 2052 2053
OR0 OR1 OR2 OR3
207
208
2083 2070 2071 2072 2073
2082
2081
2080
2ALU 2ALU 2ALU 2ALU
AND0 AND1 AND2 AND3
209
演算ユニット
2090 2091 2092 2093
OR
210
200
レジスタファイル

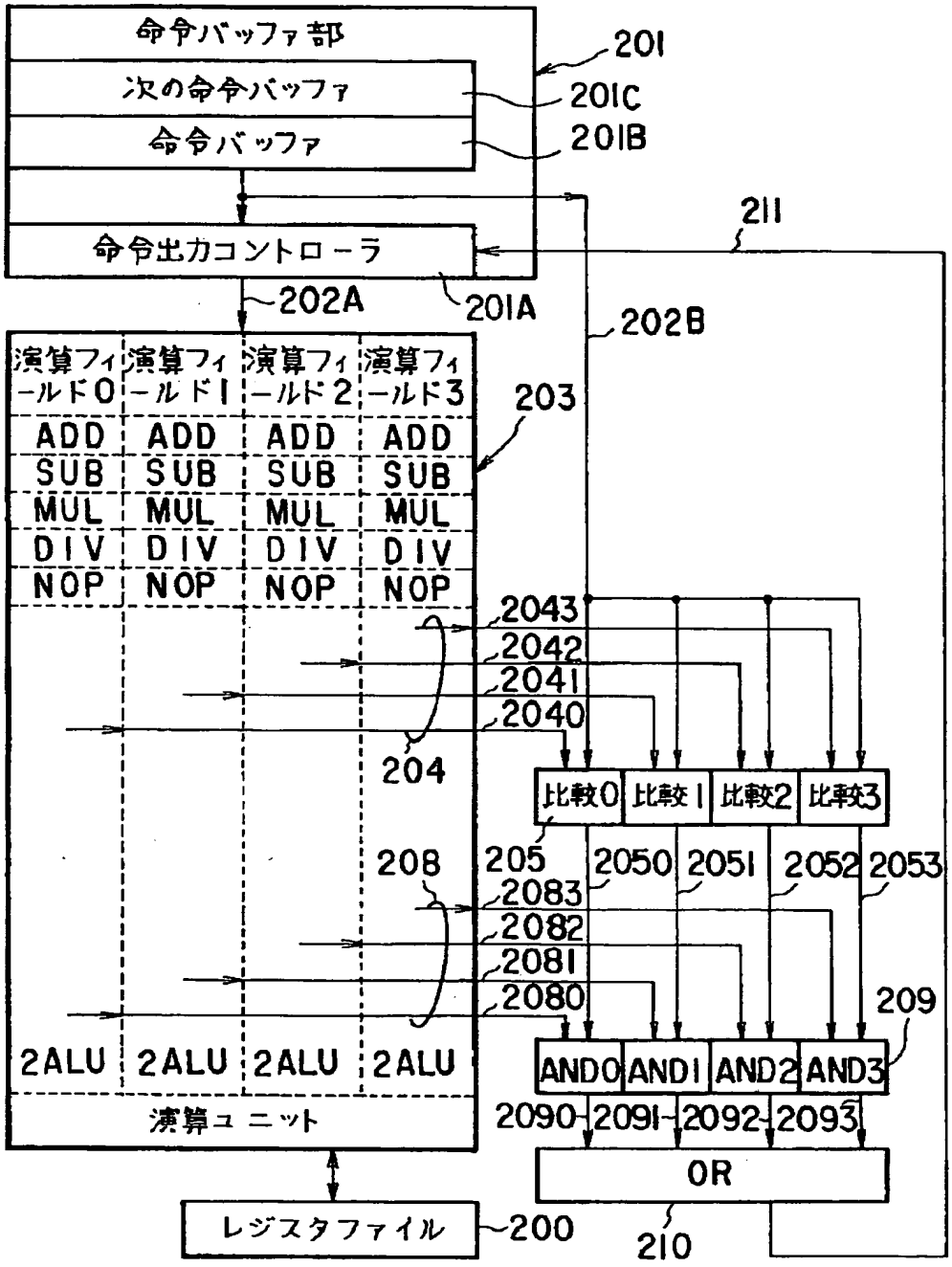
【図8】


スタート
VLIW命令フェッチ
ST10
フィールドiで現在実行中の演算結果が格納されるオペランドと、次に実行しようとするVLIW命令の全フィールドの各オペランドとを比較
ST12
一致?
ST14
NO
YES
ST16
一致のあった対応出力205iを"1"にセット
ST18
一致のなかった対応出力205iを"0"にセット
次に実行しようとするフィールドiの命令チェック
ST20
NOP?
ST22
NO
YES
ST24
NOPのあった対応出力206iを"0"にセット
ST26
NOPのなかった対応出力206iを"1"にセット
ST28
出力205iと出力206iのいずれかに"1"があるか?
NO
2
YES
1

【図９】


1
出力207iの対応部分を"1"にセット
ST30
ST32
フィールド1で現在実行中の演算がそのサイクルでまだ終了していないか
NO
YES
ST34
対応出力208iを"1"にセット
ST36
出力207iと出力208iのいずれかに"1"があるか
NO
YES
ST38
出力209iの対応部分を"1"にセット
ST40
出力209iの少なくとも一部分が"1"か
NO
YES
出力211を"1"にセット
ST42
ST44
出力208iの全てが"0"か
NO
2
YES
出力211を"0"にセット
ST46
次の命令フェッチ
ST48
リターン

【図１０】


命令バッファ部
201
次の命令バッファ
201C
命令バッファ
201B
211
命令出力コントローラ
202A
201A
202B
演算フィールド0
演算フィールド1
演算フィールド2
演算フィールド3
203
ADD
SUB
MUL
DIV
NOP
2043
2042
2041
2040
204
比較0
比較1
比較2
比較3
208
205
2050
2051
2052
2053
2083
2082
2081
2080
209
2ALU
演算ユニット
AND0
AND1
AND2
AND3
2090
2091
2092
2093
OR
レジスタファイル
200
210